Typical test task for SDET

Table of Contents

0. Preconditions

0.1 Create a new GitHub repository

0.2 Clone the new repository to your PC

0.3 Initialize the new npm project

CLI=> ***npm init -y***

0.4 Install cypress 9.6.0

CLI=> ***npm i*** cypress@9.6.0

0.5 Change the default run script that opens cypress GUI

In file package.json

```
"scripts": {
   "test": "npx cypress open"
 },
```

After that action you can start cypress GUI just typing ***npm test*** in the terminal

0.6 Change the default cypress settings

In file cypress.json add base url, skip video recording, increase the timeout, and ignore examples specs

```
{
   "baseUrl":"https://www.shopltk.com/",
   "video":false,
   "defaultCommandTimeout":7000,
   "ignoreTestFiles":["**/1-getting-started/*","**/2-advanced-examples/*"]
}
```

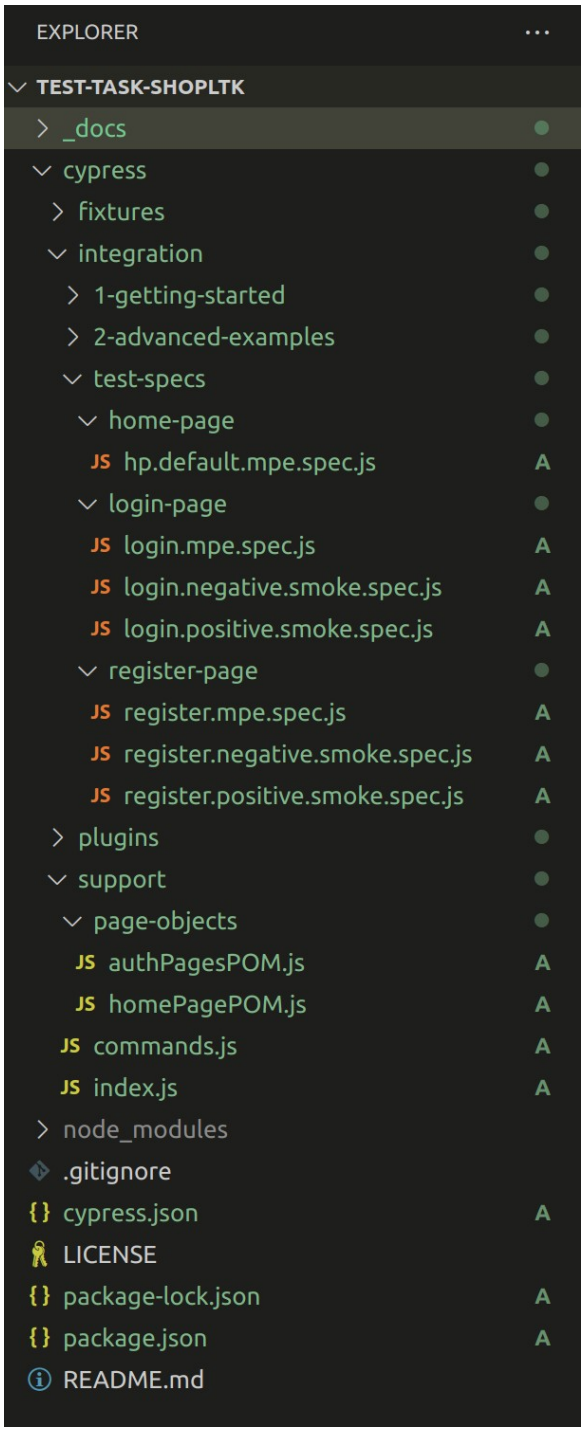
0.7 Change the default cypress settings in cypress/support/index.js

```
afterEach(() => {
  //Code to Handle the Sesssions in cypress.
  //Keep the Session alive when you jump to another test
  let str = [];
  cy.getCookies().then((cook) => {
    cy.log(cook);
    for (let l = 0; l < cook.length; l++) {
      if (cook.length > 0 && l == 0) {
        str[l] = cook[l].name;
        Cypress.Cookies.preserveOnce(str[l]);
      } else if (cook.length > 1 && l > 1) {
        str[l] = cook[l].name;
        Cypress.Cookies.preserveOnce(str[l]);
      }
    }
  });
});
```

1. Project architecture

Create a new folder **tets-specs** (inside ./cypress/integration/) - to store all the specs for the project.
Inside this folder I'm going to create a separate folder for each main page (home, register, login and so on).
Inside each these folders I'm going to create separate files for each type of tests: main page elements and their base functionality, smoke tests (positive and negative), detailed tests that check each requirement.

Create a new folder **page-objects** (inside ./cypress/support/) - to store all the Page Object Models for the project.



2. Home Page

homePagePOM.js M

cypress > support > page-objects > homePagePOM.js > HomePage > load

```
class HomePage {
  //DEFAUL PAGE - before login
  testData = {
    homePage: {
      default: {
        pageTitle: {
          selector: 'h1',
          text: 'Shop thousands of products tried and styled by real people.',
        },
        pageText: {
          text: 'Shop products across fashion, beauty, home, fitness and more. Tried and styled by real people for real life.',
        },
        signUpBtn: {
          selector: 'a[href="/signup"]',
          text: 'Sign up',
        },
        logInBtn: {
          selector: 'a[href="/login"]',
          text: 'Log In',
        },
      },
      login: {
        pageTitle: {
          selector: 'h2',
          text: 'Discover',
        },
      },
    },
  };
  //SELECTORS
  homePageDefaultTitle = () =>
    cy.get(this.testData.homePage.default.pageTitle.selector);
  signUpBtn = () => cy.get(this.testData.homePage.default.signUpBtn.selector);
  logInBtn = () => cy.get(this.testData.homePage.default.logInBtn.selector);
  //METHODS
  //Load Home Page (and check that the Home page was loaded: checking Page Title and Page Text)
  load = () => {
    cy.visit('/').location('pathname').should('eq', '/');
    cy.get(this.testData.homePage.default.pageTitle.selector)
      .should('be.visible')
      .should('have.text', this.testData.homePage.default.pageTitle.text);
    cy.get(this.testData.homePage.default.pageTitle.selector)
      .next()
      .should('be.visible')
      .should('have.text', this.testData.homePage.default.pageText.text);
    cy.get(this.testData.homePage.default.signUpBtn.selector)
      .first()
      .should('be.visible')
      .should('have.text', this.testData.homePage.default.signUpBtn.text);
    cy.get(this.testData.homePage.default.signUpBtn.selector)
      .first()
      .should('be.visible')
      .should('have.text', this.testData.homePage.default.signUpBtn.text);
  ////////////////////////////////////////////////
  //AFTER LOGIN
  homePageTitle = () => cy.get(this.testData.homePage.login.pageTitle.selector);
  userMenuButton = () => {
    cy.get('button[aria-label="menu"]');
  };
}
export default new HomePage();
```

hp.default.mpe.spec.js A

cypress > integration > test-specs > home-page > hp.default.mpe.spec.js > ...

```
import homePagePOM from '../../../support/page-objects/homePagePOM';

describe('Home Page Default (before Log In) - Main Page Elements and their base functionality Main Test Case', () => {
  describe('PRECONDITIONS: Load Home Page', () => {
    it('Visit Home Page and check that the page was loaded', () => {
      homePagePOM.load();
    });
  });
});

```

3. Login Page

login.mpe.spec.js A

cypress > integration > test-specs > login-page > login.mpe.spec.js > describe('Login Page - Main Page Elements and their base functionality Main Test Case') callback

```
import authPagesPOM from '../../../support/page-objects/authPagesPOM';
import homePagePOM from '../../../support/page-objects/homePagePOM';
authPagesPOM;
describe('Login Page - Main Page Elements and their base functionality Main Test Case', () => {
  const loginTitleText = authPagesPOM.testData.AuthPages.pageTitle.textLogin;
  const registerTitleText =
    authPagesPOM.testData.AuthPages.pageTitle.textRegister;
  const loginTextText = authPagesPOM.testData.AuthPages.pageText.loginText;
  const paragraphText = authPagesPOM.testData.AuthPages.paragraph.loginText;
  describe('PRECONDITIONS: Load Home Page', () => {…
  });
  describe('Checking Page Title', () => {
    it('Page Title should exist and be visible', () => {
      authPagesPOM.authPagesTitle().should('be.visible');
    });
    it(`Page Title should have text "${loginTitleText}"`, () => {
      authPagesPOM.authPagesTitle().should('have.text', loginTitleText);
    });
  });
  describe('Checking Page Text', () => {
    it('Page Text should exist and be visible', () => {
      authPagesPOM.authPagesText().next().should('be.visible');
    });
    it(`Page Text should have text "${loginTextText}"`, () => {
      authPagesPOM.authPagesText().should('have.text', loginTextText);
    });
  });
  describe('Checking Paragraph', () => {
    it('Page Paragraph should exist and be visible', () => {
      authPagesPOM.authPagesText().next().should('be.visible');
    });
    it(`Page Text should have text "${paragraphText} ${authPagesPOM.testData.AuthPages.signUpLink.text}"`, () => {
      authPagesPOM
        .authPagesParagraph()
        .should(
          'have.text',
          paragraphText + ' ' + authPagesPOM.testData.AuthPages.signUpLink.text
        );
    });
  });
  describe('Checking Sign Up link', () => {
    it('Sign Up link should exist and be visible', () => {
      authPagesPOM.signUpLink().should('be.visible');
    });
    it(`Sign Up link should have text "${authPagesPOM.testData.AuthPages.signUpLink.text}"`, () => {
      authPagesPOM
        .signUpLink()
        .should('have.text', authPagesPOM.testData.AuthPages.signUpLink.text);
    });
    it(`Sign Up link should redirects user to the Register Page`, () => {
      authPagesPOM.signUpLink().click();
      authPagesPOM
        .authPagesTitle()
        .should('have.text', registerTitleText)
        .should('be.visible')
        .go('back');
    });
  });
  describe('Checking Input Fields', () => {
    it('Checking Email Input Field', () => {
      authPagesPOM.checkInputField(49, 'email');
    });
    it('Checking Password Input Field', () => {
      authPagesPOM.checkInputField(52);
    });
  });
  describe('Checking Submit Button', () => {…
  });
});
```

JS login.negative.smoke.spec.js A

cypress > integration > test-specs > login-page > JS login.negative.smoke.spec.js > describe('Login Page - SMOKE TEST NEGATIVE (attem

```
1  import authPagesPOM from '../../../support/page-objects/authPagesPOM';
2  import homePagePOM from '../../../support/page-objects/homePagePOM';
3  authPagesPOM;
4  describe('Login Page - SMOKE TEST NEGATIVE (attempt to Log In with random valid credentials (
5    const emailForRegister = authPagesPOM.generateEmail();
6    const passwordForRegister = authPagesPOM.generatePassword();
7    describe('PRECONDITIONS: Load Home Page', () => {
8      it('Visit Home Page and check that the page was loaded', () => {
9        homePagePOM.load();
10     });
11     it('Click Log In button', () => {
12       homePagePOM.logInBtn().click();
13     });
14   });
15   describe('Log In action', () => {
16     it('Enter a valid email', () => {
17       cy.get(authPagesPOM.testData.AuthPages.emailInputField.fieldSelector)
18         .type(emailForRegister)
19         .should('have.value', emailForRegister);
20     });
21     it('Enter a valid password', () => {
22       cy.get(authPagesPOM.testData.AuthPages.passwordInputField.fieldSelector)
23         .type(passwordForRegister)
24         .should('have.value', passwordForRegister);
25     });
26     it('Click continue button', () => {
27       cy.get(authPagesPOM.testData.AuthPages.submitBtn.selector).click();
28     });
29     it('Checking the error message', () => {
30       cy.get(authPagesPOM.testData.AuthPages.errorMessage.selector)
31         .children('p')
32         .should(
33           'have.text',
34           authPagesPOM.testData.AuthPages.errorMessage.logintext
35         );
36     });
37   });
38 });
39
```

JS login.positive.smoke.spec.js A

cypress > integration > test-specs > login-page > JS login.positive.smoke.spec.js > ...

```
1  import authPagesPOM from '../../../support/page-objects/authPagesPOM';
2  import homePagePOM from '../../../support/page-objects/homePagePOM';
3  authPagesPOM;
4  describe('Login Page - SMOKE TEST POSITIVE - Main Test Case', () => {
5    const emailForLogin = authPagesPOM.testData.users.test1.email;
6    const passwordForLogin = authPagesPOM.testData.users.test1.password;
7    describe('PRECONDITIONS: Load Home Page', () => {
8      it('Visit Home Page and check that the page was loaded', () => {
9        homePagePOM.load();
10     });
11
12     it('Click Log In button', () => {
13       homePagePOM.logInBtn().click();
14     });
15   });
16   describe('Log In action', () => {
17     it('Enter a valid email (that was registered before)', () => {
18       cy.get(authPagesPOM.testData.AuthPages.emailInputField.fieldSelector)
19         .type(emailForLogin)
20         .should('have.value', emailForLogin);
21     });
22     it('Enter a valid password (that was registered before)', () => {
23       cy.get(authPagesPOM.testData.AuthPages.passwordInputField.fieldSelector)
24         .type(passwordForLogin)
25         .should('have.value', passwordForLogin);
26     });
27     it('Click continue button', () => {
28       cy.get(authPagesPOM.testData.AuthPages.submitBtn.selector).click();
29     });
30   > it('Checking that user was logged in successfuly. User profile button should a
38     });
39     it('Log Out', () => {
40       authPagesPOM.logOut();
41     });
42   });
43 });
44
```

4. Register Page

register.mpe.spec.js A

cypress > integration > test-specs > register-page > register.mpe.spec.js > describe('Login Page - Main Page Elements and their base functionality Main Test Case') callback >

```
import authPagesPOM from '../../../support/page-objects/authPagesPOM';
import homePagePOM from '../../../support/page-objects/homePagePOM';
authPagesPOM;
describe('Login Page - Main Page Elements and their base functionality Main Test Case', () => {
  const registerTitleText =
    authPagesPOM.testData.AuthPages.pageTitle.textRegister;
  const loginTitleText = authPagesPOM.testData.AuthPages.pageTitle.textLogin;
  const registerTextText =
    authPagesPOM.testData.AuthPages.pageText.registerText;
  const paragraphText = authPagesPOM.testData.AuthPages.paragraph.registerText;
  describe('PRECONDITIONS: Load Home Page', () => {
    it('Visit Home Page and check that the page was loaded', () => {
      homePagePOM.load();
    });
    it('Click Log In button', () => {
      homePagePOM.signUpBtn().click();
    });
  });
  describe('Checking Page Title', () => {
    it('Page Title should exist and be visible', () => {
      authPagesPOM.authPagesTitle().should('be.visible');
    });
    it(`Page Title should have text "${registerTitleText}"`, () => {
      authPagesPOM.authPagesTitle().should('have.text', registerTitleText);
    });
  });
  describe('Checking Page Text', () => {
    it('Page Text should exist and be visible', () => {
      authPagesPOM.authPagesText().next().should('be.visible');
    });
    it(`Page Text should have text "${registerTextText}"`, () => {
      authPagesPOM.authPagesText().should('have.text', registerTextText);
    });
  });
  describe('Checking Paragraph', () => {
    it('Page Paragraph should exist and be visible', () => {
      authPagesPOM.authPagesText().next().should('be.visible');
    });
    it(`Page Text should have text "${paragraphText} ${authPagesPOM.testData.AuthPages.logInLink.text}"`, () => {
      authPagesPOM
        .authPagesParagraph()
        .should(
          'have.text',
          paragraphText + ' ' + authPagesPOM.testData.AuthPages.logInLink.text
        );
    });
  });
  describe('Checking Log In link', () => {
    it('Log In link should exist and be visible', () => {
      authPagesPOM.logInLink().should('be.visible');
    });
    it(`Log In link should have text "${authPagesPOM.testData.AuthPages.logInLink.text}"`, () => {
      authPagesPOM
        .logInLink()
        .should('have.text', authPagesPOM.testData.AuthPages.logInLink.text);
    });
    it(`Log In link should redirects user to the Register Page`, () => {
      authPagesPOM.logInLink().click();
      authPagesPOM
        .authPagesTitle()
        .should('have.text', loginTitleText)
        .should('be.visible')
        .go('back');
    });
  });
  describe('Checking Input Fields', () => {
    it('Checking Email Input Field', () => {
      authPagesPOM.checkInputField(49, 'email');
    });
    it('Checking Password Input Field', () => {
      authPagesPOM.checkInputField(55);
    });
  });
  describe('Checking Submit Button', () => {
    it('Checking that Submit Button exists', () => {
      authPagesPOM.submitButton();
    });
    it(`Checking that Submit Button has text "${authPagesPOM.testData.AuthPages.submitBtn.registerText}"`, () => {
      authPagesPOM
        .submitButton()
        .should(
          'have.text',
          authPagesPOM.testData.AuthPages.submitBtn.registerText
        );
    });
    it('Checking that Submit Button is clickable', () => {
      authPagesPOM.submitButton().click();
    });
  });
});
```

register.negative.smoke.spec.js A

cypress > integration > test-specs > register-page > register.negative.smoke.spec.js > describe('Register Page - SMOKE TEST NEGATIVE (atempt to

```
1  import authPagesPOM from '../../../support/page-objects/authPagesPOM';
2  import homePagePOM from '../../../support/page-objects/homePagePOM';
3  authPagesPOM;
4  describe('Register Page - SMOKE TEST NEGATIVE (atempt to register a new user with email that was alrea
5    const emailForLogin = authPagesPOM.testData.users.test1.email;
6    const passwordForLogin = authPagesPOM.testData.users.test1.password;
7    describe('PRECONDITIONS: Load Home Page', () => {
8      it('Visit Home Page and check that the page was loaded', () => {
9        homePagePOM.load();
10     });
11     it('Log Out', () => {});
12     it('Click Sign Up button', () => {
13       homePagePOM.signUpBtn().click();
14     });
15   });
16   describe('Sign Up action', () => {
17     it('Enter a valid email (that was registered before)', () => {
18       cy.get(authPagesPOM.testData.AuthPages.emailInputField.fieldSelector)
19         .type(emailForLogin)
20         .should('have.value', emailForLogin);
21     });
22     it('Enter any valid password', () => {
23       authPagesPOM
24         .getInputById(20)
25         .type(passwordForLogin)
26         .should('have.value', passwordForLogin);
27     });
28     it('Click continue button', () => {
29       cy.get(authPagesPOM.testData.AuthPages.submitBtn.selector).click();
30     });
31     it('Checking the error message', () => {
32       cy.get(authPagesPOM.testData.AuthPages.errorMessage.selector)
33         .children('p')
34         .should(
35           'have.text',
36           authPagesPOM.testData.AuthPages.errorMessage.registerText
37         );
38     });
39   });
40 });
41
```

register.positive.smoke.spec.js A

cypress > integration > test-specs > register-page > register.positive.smoke.spec.js > describe('Register Page - SMOKE TEST POSITIVE - Main Te

```
1  import authPagesPOM from '../../../support/page-objects/authPagesPOM';
2  import homePagePOM from '../../../support/page-objects/homePagePOM';
3  authPagesPOM;
4  describe('Register Page - SMOKE TEST POSITIVE - Main Test Case', () => {
5    const emailRegistration = authPagesPOM.generateEmail();
6    const passwordForRegistration = authPagesPOM.generatePassword();
7    describe('PRECONDITIONS: Load Home Page', () => {
8      it('Visit Home Page and check that the page was loaded', () => {
9        homePagePOM.load();
10     });
11     it('Click Sign Up button', () => {
12       homePagePOM.signUpBtn().click();
13     });
14   });
15   describe('Sign Up action', () => {
16     it('Enter a valid email (that was registered before)', () => {
17       cy.get(authPagesPOM.testData.AuthPages.emailInputField.fieldSelector)
18         .type(emailRegistration)
19         .should('have.value', emailRegistration);
20     });
21     it('Enter any valid password', () => {
22       authPagesPOM
23         .getInputById(20)
24         .type(passwordForRegistration)
25         .should('have.value', passwordForRegistration);
26     });
27     it('Click continue button', () => {
28       cy.get(authPagesPOM.testData.AuthPages.submitBtn.selector).click();
29     });
30 >   it('Checking that user was registered successfuly. And redirected to the Home Page', () => {…
38     });
39     it('Log Out', () => {
40       authPagesPOM.logOut();
41     });
42   });
43 });
44
```

5. Run Scripts

package.json > ...

```
{
  "name": "test-task-shopltk",
  "version": "1.0.0",
  "description": "Example of test task",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "npx cypress open",
    "mpe":"npx cypress run --spec 'cypress/integration/test-specs/*/*mpe.spec.js'",
    "smokePositive":"npx cypress run --spec 'cypress/integration/test-specs/*/*positive.smoke.spec.js'",
    "smokeNegative":"npx cypress run --spec 'cypress/integration/test-specs/*/*negative.smoke.spec.js'",
    "regression":"npm run mpe && npm run smokePositive && npm runNegative"


  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ILopatenko/test-task-shopltk.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ILopatenko/test-task-shopltk/issues"
  },
  "homepage": "https://github.com/ILopatenko/test-task-shopltk#readme",
  "dependencies": {
    "cypress": "^9.6.0"
  }
}

```

6. gitHub Actions pipeline

29 lines (27 sloc) | 803 Bytes

```yaml
name: Main Pipe Line

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [16.x]
    steps:
    - uses: actions/checkout@v3
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v3
      with:
        node-version: ${{ matrix.node-version }}
        cache: 'npm'
    - name: Stage 0. PRECONDITIONS. Download all the 3rd party packages using NPM I command
      run: npm i
    - name: Stage 1. Testing Main Pages and Their Base Functionality
      run: npm run mpe
    - name: Stage 2. Smoke Tests (Happy Path) for all the pages
      run: npm run smokePositive
    - name: Stage 3. Smoke Tests (Negative) for all the pages
      run: npm run smokeNegative
```